ECOSYSTEM MEETING, April 9th, 2008

Why to use RDF, not XML nor relation model ?

- we will execute relation-directed queries, e.g. : *What are inmovable the devices in the same locations, where person P was observed during last N seconds ?*
- when mapping relational model to RDF one, then the table can be understood as ontological class and column names in the table – as properties, and records – as instances of the class [1]
- relational model better for queries against data in a single table (about a single class of objects) ; joins on tables are time- and resource-consuming
- hierarchical structure of XML flats incovient for quering loosely coupled nodes; Xpath query requires traversings to the interesting node along long paths, often without interesting information on a way
- RDF is focused on re-usable relationships and better for pattern matching queries; joins on table are more natural
- SQL query returns results are in tabular form
- SPARQL query can return result not only in tabular form (SELECT-like query), but also as sub-graphs [2]:
  - rooted at the certain resource/node (DESCRIBE-like query), e.g.: *What inmovable devices are in the location L ?* lead us to the question what do we want to know exactly about the devices, as device ID seems not satisfying
  - constructed according to the query requirements (CONSTRUCT-like query)
- these sub-graphs can be later locally processed and connected with other information

Literature:

1. Tim Berners-Lee, *Relational Databases on the Semantic Web*, http://www.w3.org/DesignIssues/RDB-RDF.html
2. *W3C Recommendation: SPARQL Query Language for RDF, Query Forms*, http://www.w3.org/TR/rdf-sparql-query/#QueryForms
3. Benjamin Szekely, Elias Torres, *A Semantic Data Collection Model for Sensor Network Applications*, http://www.klinewoods.com/papers/semanticdcn.pdf

How to store, access and process RDF data

- RDF data can processed by the machine as one or more RDF graphs
- RDF data can be stored in a temporal way (in-memory data models), or a persistent way (database / file with XML-like or N3 serialized RDF / native file)
- We would like to read/write data remotely and also process them locally
- RDF data can be queried with different query languages: RDQL, SeRQL, SPARQL. We choose SPARQL which has W3C recommendation.
- Most prominent semantic frameworks, I've also worked with are: Jena 2.x and Sesame 2.x. Both provide tools for storing data in a temporal and persistent ways, update and query them with SPARQL queries, both locally and remotely.
- Some examples of access to data: Jena – in Eclipse, Sesame – http://localhost:8080/openrdf-workbench/

Literature:

1. *Jena Semantic Framework*, http://jena.sourceforge.net/
2. *Jena RDB (Persistent RDF data in relational databases)*, http://jena.sourceforge.net/documentation.html
3. *ARQ - A SPARQL Processor for Jena*, http://jena.sourceforge.net/ARQ/
4. *Chapter 6 of Sesame documentation: The SeRQL query language*, http://www.openrdf.org/doc/sesame/users/ch06.html
5. *W3C Member Submission: RDQL - A Query Language for RDF*, http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
6. *Sesame: RDF Schema Querying and Storage*, http://www.openrdf.org/
7. *Jena SDB*, http://jena.hpl.hp.com/wiki/SDB

How to specify conceptualization of knowledge gathered from sensor networks

- Localisation issues
- Working with examples (paper + Sesame Web interface)

Literature:

1. David J. Russomanno, Cartik R. Kothari and Omoju A. Thomas, *Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models*, http://www.engr.memphis.edu/eece/cas/publications/ica3194.pdf
2. Thibaud Flury, Gilles Privat, Fano Ramparany, *OWL-based location ontology for context-aware services*, http://w5.cs.uni-sb.de/~baus/aims04/cameraready/P8.pdf

Performance issues

- allarming number of factors which matters: the hardware, the operating system, the database engine and its myriad parameters, the data itself (RDF graph sparsity, blank nodes etc.), the queries (number of unbound variables vs. bound ones, number of relations etc.), inferencing turned of and its type, and planetary alignment ;-) [1]
- engines efficiency depends on the type of operation: building hashes/indexes during loading data into a store is space- and time-consuming, but results in great impovement, when quering the data [1]
- examples [2]: (focus on Jena SDB/PostgreSQL and Sesame Native File/PostgreSQL) and [4] (Jena RDB/PostgreSQL and Jena SDB/PostgreSQL)

Literature:

1. *SDB/Loading performance*, http://jena.hpl.hp.com/wiki/SDB/Loading_performance
2. *Benchmarks for Sesame Native Repository and Jena SDB*, http://www4.wiwiss.fu-berlin.de/benchmarks-200801/
3. *RDF Store Benchmarks list*, http://esw.w3.org/topic/RdfStoreBenchmarking
4. http://jena.hpl.hp.com/wiki/SDB/Query_performance

Usability opinions

1. Jena 2.x
   - *very clear and legible documentation!*
   - *Great, active mailing list*
   - *very intuitive API*
2. Sesame 1.x
   - *documentation appears to be legible, but many aspects are touched so toroughly, so when you sit down to pogramming, then it comes out, then one operation can be implemented in 1000 unmentioned ways*
   - *quite food forum: the activity is high, but reponses are not so complete and extensive as in Jena mailing list*
   - *API misty, similary to documentation: planning an application requires first to check different trick with it on a side*
3. Sesame 2.x
   - *different documentation for administrator and user*
   - *many problems on forum has been replied with sentence: switch to Sesame 2.x – there are no such problems as in Sesame 1.x*
   - *API seems clearer then in Sesame 1*
   - *imposible to use with J2ME, as it uses Java 1.5 generic types*

(thanks for help to Paweł Skornicz from the Wroclaw University of Technology)

Learning and working with RDF in detail

- I think you can also find some things in italian, but here is the stuff I could find useful for me.
- Theory:
  - I would start with simple examples what RDF is at [1],
  - and the fact XML is one of the way of representing RDF, among others (RDF/XML, RDF/XML-ABBREV, RDF/N3, RDF/NTriples) [2]
  - and, some general idea of founder of WWW and Semantic Web, Tim Berners-Lee [3]
- Practice:
  - I would start concurrently playing with some software which can handle RDF. I mainly use Jena (open source from IBM), avaialable here [4]
  - They provided a nice tutorial to start with [5],
  - Having that I would start with playing with tools and quering RDF data with some query language (most common is SPARQL).
- Designing:
  - I left it for the end, 'cause I believe it's sometimes better to start from pure RDF sources instead of designing RDF ontologies in WYSIWYG apps. I recommend you quite stable and free app from Standford [7],
  - and some tutorials for it [8].

Literature:

1. *Primer: Getting into RDF & Semantic Web using N3*, http://www.w3.org/2000/10/swap/Primer
2. *Comparing Formats*, http://www.w3.org/2000/10/swap/doc/formats
3. *An readable language for data on the Web*, http://www.w3.org/DesignIssues/Notation3.html
4. *Jena, Semantic Framework*, http://jena.sourceforge.net
5. *Jena Tutorial*, http://jena.sourceforge.net/tutorial/
6. *Search RDF data with SPARQL*, http://www.ibm.com/developerworks/java/library/j-sparql/
7. *The Protege Ontology Editor and Knowledge Acquisition System*, http://protege.stanford.edu/
8. *Getting Started with Protege-Frames*, http://protege.stanford.edu/doc/tutorial/get_started/table_of_content.html